# Duke Nukem 3D - Ein Klassiker in neuem Gewand: EDuke32 im Eigenbau unter Linux

Dr. Christian Behrens

## Inhalt



## Einleitung

Duke Nukem, Protagonist des nach ihm benannten Ego-Shooters Duke Nukem 3D – oder kurz: Duke 3D – war ursprünglich im Grunde ein äußerst friedfertiger, mitfühlender und diskussionsbereiter Zeitgenosse. Als aber feindliche Aliens seine „Chicks“ entführten, diskutierte er nicht lange mit ihnen: Er trat sie in den Ar*** und andere Körperteile, die diese Bastarde wahrscheinlich gar nicht hatten und… - das war es dann eigentlich schon mit der Handlung des Spiels, das damals wie heute, wenn auch aus jeweils anderen Gründen, polarisieren dürfte – gut so!

Tatsächlich handelt es sich beim „Duke“ um eine Art Kreuzung aus Arnold Schwarzenegger (in Predator) und Sylvester Stallone (in Rambo II), also einen aus der Zeit gefallenen Anachronismus. Trotzdem – oder genau deswegen – hat es sehr viel Spaß gemacht, sich diesen Helden meiner Jugend noch einmal vorzunehmen. Ursprünglich bereits 1996 veröffentlicht, schien es mir allerdings wenig sinnvoll, den großartigen, aber doch etwas in die Jahre gekommenen „3D-Ego-Shooter“, ein Computerspiel, das wirklich alles war – nur nicht jugendfrei oder gar politisch korrekt – auf einer zeitgenössischen Maschine zu spielen. Selbstverständlich ist das möglich, unter Linux z. B. in der DosBOX, macht aber bereits angesichts der niedrig aufgelösten Texturen nicht mehr wirklich Spaß, denn das sähe alles doch arg verpixelt aus. Genau diese Lücke haben – wie auch bei anderen Klassikern der 90er Jahre – engagierte und talentierte Fans geschlossen, die nicht nur neue „Engines“ für den Duke programmierten, sondern nicht zuletzt eine Vielzahl feiner, hoch auflösender Grafiken für das Spiel erstellten. Damit macht es dann sogar in Full-HD auf einem (sehr) großen Bildschirm wieder Spaß.

Der folgende Beitrag stellt kurz die Geschichte (I.), den Inhalt und vor allem die Besonderheiten des Spiels (II.) dar. Ein Tutorial bzw. eine Anleitung erklärt Schritt für Schritt, wie das Spiel in neuem Gewand unter Linux zum Laufen gebracht und erweitert werden kann (III.).

# I. Duke Nukem 3D – Ein Klassiker im Rückblick

Nachdem Computerspiele wie Wolfenstein 3D oder vor allem Doom angesichts ihrer für damalige Verhältnisse revolutionären, zumindest nach 3D aussehenden Grafik für Furore gesorgt hatten, veröffentlichte die Apogee Software, Ltd. bzw. (später) 3D Realms Anfang 1996 Duke Nukem 3D. Nüchtern technisch betrachtet tatsächlich eher 2,5D; aber es sah schon verdammt nach 3D aus, zumal die Engine anders als z. B. die von Doom auch schräge Flächen darstellen konnte. Auf der Grundlage erstellte u. a. der bekannte Leveldesigner Richard „Levelord“ Gray sehr ausgefeilte Level, die wesentlich mit zum hohen Spielspaß beitrugen.

Wegen seiner doch recht expliziten Gewaltdarstellungen (...) landete das Spiel prompt auf dem „Index“. Erst Anfang 2017 (sic!) hob die Bundesprüfstelle die Indizierung wieder auf.

Bereits 2003 (am 1. April) veröffentliche 3D Realms den Quelltext des Spieles unter der GNU GPL, schloss den Spielinhalt aber davon aus.

# II. Das Spiel und seine Besonderheiten (Achtung: Spoiler!)

Die Handlung – soweit man davon sprechen kann – des Spiels ist schnell beschrieben: Feindliche Aliens überfallen die Erde und entführen u. a. die „Chicks“ des Duke. Grund genug für ihn, die Sache selbst in die Hand zu nehmen und zum Gegenangriff überzugehen. Das war es im Grunde.

Das Spiel enthält zahlreiche popkulturelle Verweise etwa auf Filmklassiker wie Alien – Das unheimliche Wesen aus einer fremden Welt oder Aliens – Die Rückkehr. Wer sich erstmals mit den an Facehugger erinnernden Aliens herumplagen darf oder die in der Nähe von Alieneiern „eingesponnenen“ Frauen zu sehen bekommt, wird wissen, was ich meine:

Beschränkte sich Duke Nukem 3D allerdings darauf, wäre das Spiel längst vergessen – und das völlig zu Recht. Was macht es also so besonders, was sorgt für den zeitlosen Spielspaß?

*Achtung – Spoilerwarnung: Wer das Spiel noch nicht kennt und sich die Überraschung nicht verderben lassen möchte, sollte den folgenden Text überspringen und unter* ***III. Tutorial: EDuke32 im Eigenbau unter Linux*** *weiterlesen.*

Zuerst einmal das Design der durchgehend sehr ausgefeilten Level des Spiels: Bis heute ist es mir nicht gelungen, alle „Secrets“, also geheimen Bereiche der Level zu entdecken.

Einige Level verändern während des Spielens nach entsprechenden Handlungen des Spielers sogar ihre Struktur, so sprengt der Protagonist auch gerne einmal ein ganzes Gebäude aus dem Weg usw.

Daneben sorgen technische Innovationen für nachhaltigen Spielspaß: Mittels Überwachungskameras kann der Duke einzelne, noch unbekannte Bereiche des jeweiligen Levels durch Benutzen der Monitore betrachten, dadurch künftige Gegner bereits vorab und Folgen seiner Aktionen, etwa das Betätigen eines Schalters erkennen.

Das „Jetpack“ lässt ihn fliegen, mit der Nachtsichtbrille erkennt er Gegner in der Dunkelheit usw.

Damals wie heute tragen sehr innovative Waffen ebenfalls erheblich zum Spielspaß bei: Neben obligatorischer Pistole und Schrotflinte kann der Duke schon bald auch „Pipebombs“ nutzen: Rohrbomben, die er zunächst platzieren und dann aus sicherer Entfernung zünden kann oder das zumindest tunlichst sollte, sonst – nun ja…:

Humoristischer Höhepunkt allerdings sind Alienwaffen wie „Shrinker“ und „Freezer“: Mit dem Shrinker z. B. kann Duke seine Gegner erheblich verkleinern, um sie dann schlicht zu zertreten („squish“ bzw. „crunch“):


Der Freezer vereist Gegner, die der Duke dann ebenfalls zertreten oder eher zersplittern kann.

Vor allem aber der unnachahmliche, absolut schräge und politisch ganz und gar nicht korrekte Humor des Spiels macht es auch heute noch und damit zeitlos komisch: Das beginnt schon mit den Sprüchen des Duke, der viele seiner Handlungen kommentiert – oder auch die Untätigkeit des Spielers: „What are you waiting for – Christmas?!“

Waffenfunde kommentiert er ebenso („Groovy!“, „Come get some...“) wie sein eigenes Spiegelbild („Damn – I‘m looking good!“). Toiletten im Spiel kann er tatsächlich benutzen („Ah – that‘s better...“) oder wahlweise Aliens buchstäblich vom Thron schießen, die dort ebenfalls zuweilen ihren Bedürfnissen nachgehen:

Mit Frauen, insbesondere Stripperinnen im Spiel kann der Duke interagieren: Steckt er ihnen etwas Geld zu – nun ja, seht selbst…

Ob der „Konkurrenz“ mittels gefallenem Doomtrooper nun Respekt erwiesen oder der Mittelfinger gezeigt werden sollte, kann meines Erachtens dahinstehen: So oder so nur ein weiterer lustiger Höhepunkt, als der Duke in einem der Level den Trooper aus Doom™ findet und das ebenfalls entsprechend kommentiert („That’s one doomed space marine!“):


Neben den spannenden Leveln des Spiels ist es vor allem dieser Humor, der für nachhaltigen Spielspaß sorgt. Aber genug davon – nun zum Spiel bzw. zum Tutorial:

## III. Tutorial: EDuke32 im Eigenbau unter Linux

Angesichts des hohen Unterhaltungswerts überrascht die nach wie vor anhaltende Begeisterung vieler Fans nicht wirklich. Nach Veröffentlichung des Quelltextes 2003 portierten einige das Spiel u. a. auf die Linux-Plattform.

Für meine Zwecke nutze ich unter Linux Mint 18.1 (Serena) mit EDuke32 eine quelloffene Entwicklung, die für nicht-kommerzielle Zwecke kostenlos genutzt werden darf. Die Software läuft auf den unterschiedlichsten Plattformen, neben Linux etwa unter Windows, Mac OS X oder FreeBSD – bis hin zum „family toaster“ und „your girlfriend's vibrator“ (so zumindest die Programmierer auf ihrer Website), das aber eher der Vollständigkeit halber.

*Achtung: EDuke32 ermöglicht das Spielen von Duke Nukem 3D, *ist* aber nicht das vollständige Spiel selbst, da 3D Realms sich seinerzeit die Rechte am Spielinhalt vorbehielt. Zum Spielen ist daher nach wie vor die Datei **duke3d.grp** des Originalspiels oder der Shareware-Version erforderlich. Wer keine alte Diskette oder CD mehr hat, wird evtl. auf Steam™ fündig.*

Diese Software führt das Spiel nativ aus, benötigt also keinerlei Emulator o. ä., um z. B. die DOS-Plattform des ursprünglichen Spiels nachzuahmen. Dabei verbessert EDuke32 das ursprüngliche Spiel deutlich. Nun bestehen die Gegner im Spiel aus echten 3D-Modellen statt der ursprünglichen 2D-Sprites. Duke 3D kann nun auch in „crazy resolutions like 3072x2304“ (O-Ton) gespielt werden – ganz ehrlich: In Full-HD bzw. 1920x1080 auf einem 49-Zoll-Bildschirm reichte mir das schon völlig…

Für den Eigenbau unter Linux Mint 18.1 habe ich den Quellcode heruntergeladen (1.), das Kompilieren vorbereitet (sämtliche Abhängigkeiten erfüllt usw.) (2.), die Software dann kompiliert (3.), schließlich eingerichtet (ein Installieren im eigentlichen Sinne ist nicht erforderlich), und dabei vor allem mit ein paar sinnvollen Erweiterungen ergänzt (4.). Meine Beschreibung folgt dabei im Wesentlichen der offiziellen Anleitung, die ich auf Grundlage meiner Erfahrungen damit hier noch etwas ergänzt habe:

## 1. Herunterladen und soweit erforderlich Entpacken des Quelltextes

EDuke32 wird gegenwärtig aktiv weiter entwickelt und gepflegt, daher kann bei Erscheinen dieses Textes bereits eine aktuellere Version des Quelltextes erschienen sein.

Daher empfehle ich den Download der jeweils aktuellen Version direkt aus dem „EDuke32 Subversion (SVN) Repository“. Folgende Befehle habe ich dafür im Terminal eingegeben bzw. in der Shell ausgeführt:

*Hinweis (für Linux-Neulinge): Unter Linux arbeite ich sehr viel mit dem Terminal. Dennoch tippe ich kaum Befehle ein, denn unter Linux kann markierter Text per Klick auf die mittlere Maustaste einfach in die Konsole bzw. das Terminal kopiert werden. Also: Nachstehende Befehlszeilen bitte nicht abtippen (schon gar nicht die längeren weiter unten...), sondern markieren und per Mausklick (oder notfalls via copy & paste) ins Terminal einfügen – Exkurs Ende.*

```
$ cd ~/Downloads

$ mkdir Software

$ cd Software

$ svn checkout http://svn.eduke32.com/eduke32/
```

Das sollte dann zu folgender Ausgabe führen:

```
A    eduke32/jit
A    eduke32/package
A    eduke32/package/common
A    eduke32/package/debug
```

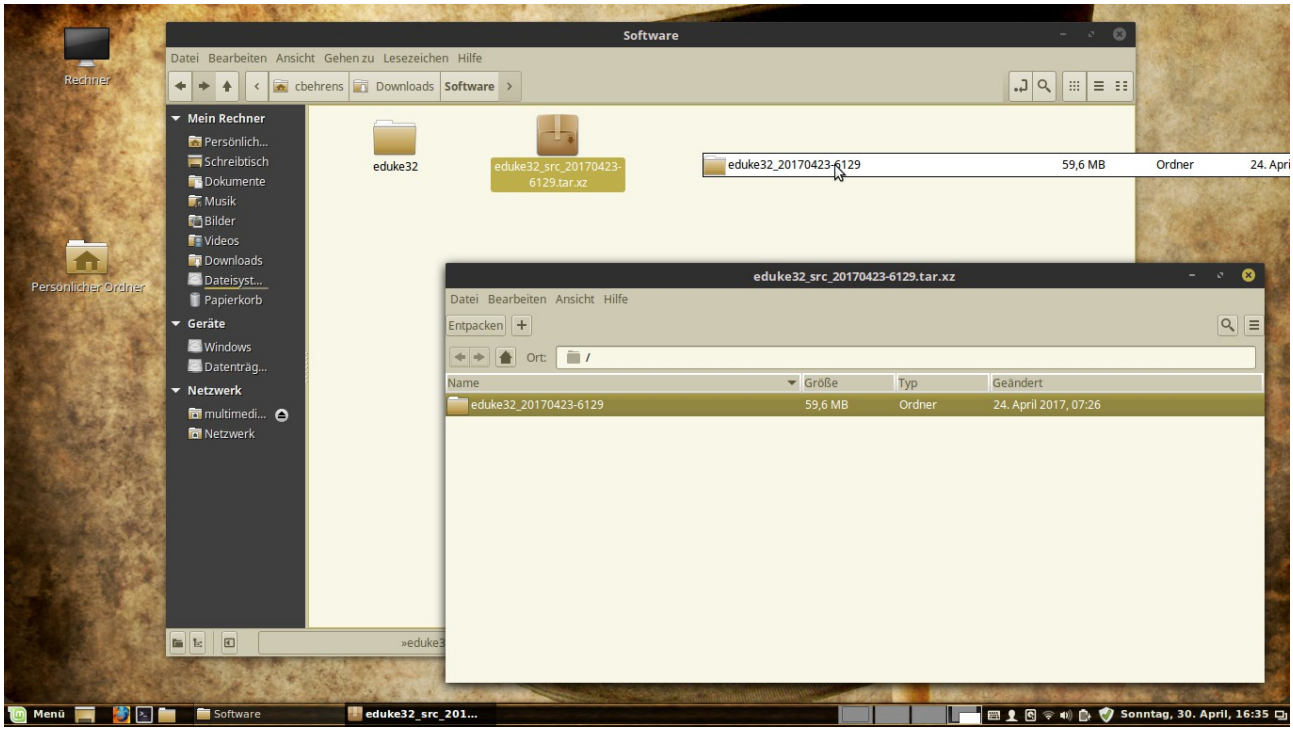
```
A    eduke32/package/debug/win32
A    eduke32/package/debug/win64
A    eduke32/package/sdk
```

[…] - *sehr* stark gekürzt, bei Interesse s. Fassung online...

```
A    eduke32/source/tools/src/wad2art.cpp
A    eduke32/Common.mak
A    eduke32/.gitignore
A    eduke32/_clang-format
A    eduke32/Android.mk
A    eduke32/source/tools/src/wad2map.cpp
A    eduke32/GNUmakefile
 U   eduke32
Ausgecheckt, Revision 6129.
```

Alternativ kann der jeweils aktuelle Quelltext (oder bei Bedarf auch ältere Versionen) als Tarball (Archiv) von der Website der Entwickler heruntergeladen werden: **eduke32_src_20170423-6129.tar.xz** (Stand: 27. April 2017).

Das Archiv habe ich ebenfalls in das Verzeichnis Software verschoben, dort mit dem Archivmanager entpackt und anschließend wieder gelöscht:




Die folgend beschriebenen Schritte gelten für alle Quelltexte, egal ob aus SVN-Repository oder Tarball bezogen gleichermaßen.

## 2. Vorbereitung

Bevor das Programm unter Linux ausgeführt werden kann, muss der Quelltext erst noch kompiliert werden. Generell muss die dafür benötigte Software (Compiler usw.) installiert sein. Unter Linux Mint ist das standardmäßig der Fall – kann mich zumindest nicht erinnern, das erst noch nachgeholt zu haben.

Daneben müssen für EDuke32 verschiedene besondere Abhängigkeiten erfüllt sein, d. h. z. B. verschiedene Programmbibliotheken („libraries“) müssen installiert sein und bei Bedarf noch zwingend installiert werden.

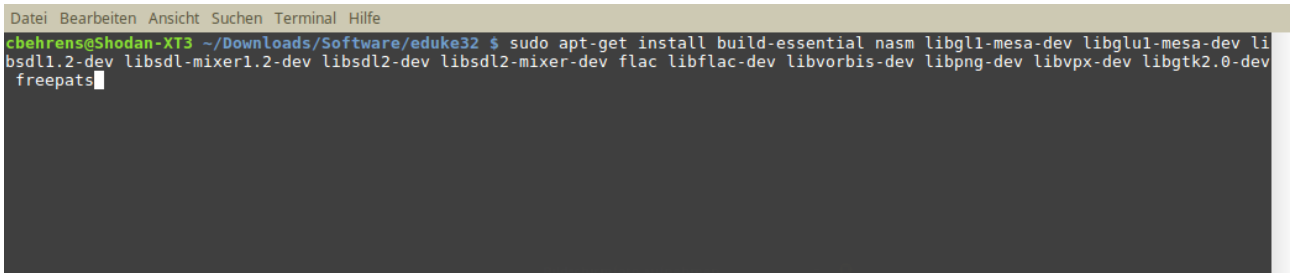
Unter Ubuntu (ab 14.04) und Derivaten, wie z. B. Linux Mint können folgende Pakete

- **build-essential**
- **nasm**
- **libgl1-mesa-dev**
- **libglu1-mesa-dev**
- **libsdl1.2-dev**
- **libsdl-mixer1.2-dev**
- **libsdl2-dev**
- **libsdl2-mixer-dev**
- **flac**
- **libflac-dev**
- **libvorbis-dev**
- **libpng12-dev**
- **libvpx-dev**
- **libgtk2.0-dev**
- **freepats**

mit dem Befehl

```
$ sudo apt-get install build-essential nasm libgl1-mesa-dev libglu1-mesa-dev
libsdl1.2-dev libsdl-mixer1.2-dev libsdl2-dev libsdl2-mixer-dev flac libflac-dev
libvorbis-dev libpng-dev libvpx-dev libgtk2.0-dev freepats
```

installiert werden:




Das Programm apt-get benötigt Administratorrechte, unter Linux Mint wird es daher mit dem vorangestellten Kommando „sudo“ ausgeführt.

Besonders möchte ich auf die Pakete **flac** und **libflac-dev** hinweisen: In der Anleitung aus dem Ubuntu-Wiki fehlen beide Pakete, was mir zunächst erhebliche Probleme bereitete, denn das Kompilieren scheiterte beim Objekt **flac.o** – allerdings mit einer zumindest für mich etwas irreführenden Fehlermeldung. Danach konnte der Kompiler (angeblich) eine definitiv vorhandene Datei (**all.h**) nicht finden, die allerdings war im Verzeichnis **~/Downloads/Software/eduke32/platform/Windows/include/FLAC** vorhanden. So etwas kann interessierte Laien wie mich natürlich verwirren, daher habe ich – eher auf Verdacht – die Pakete **flac** und l**ibflac-dev** nachinstalliert und dann lief der Kompiliervorgang ohne weitere Probleme durch.

## 3. Kompilieren der Software

Sind alle Abhängigkeiten erfüllt, ist der eigentliche Kompiliervorgang fast schon ein Selbstläufer.

Soweit wir nicht schon darin sind, wechseln wir in der Konsole in das Verzeichnis eduke32:

```
$ cd ~/Downloads/Software/eduke32
```

*Hinweis: Die Tilde „~“ steht unter Linux immer für das jeweilige Heimatverzeichnis, d. h. egal, von welchem (Unter-)Verzeichnis aus der Befehl ausgeführt wird (in meinem Beispiel im Wurzel- bzw. Rootverzeichnis): Es spart Tipperei (denn sonst hätte ich $ cd /home/cbehrens/Downloads/… eingeben müssen) und man landet immer richtig…*

```
$ ls
```

zeigt den Inhalt des Verzeichnisses an, in dem u. a. das GNUmakefile zu finden ist: Eine Datei mit einer Vielzahl von Anweisungen für Kompiler und Linker, auf die ich hier nicht näher eingehen kann.

Den eigentlichen Kompilier- und Linkvorgang starten wir nun mit dem Befehl

```
$ make
```

(Jupp – mehr ist es tatsächlich nicht…).

Der führt dann – hoffentlich – zu folgender, im Original farbiger Ausgabe (s. Screenshots online):

```
Built object obj/duke3d/game.o
Built object obj/duke3d/global.o
Built object obj/duke3d/actors.o
Built object obj/duke3d/gamedef.o
Built object obj/duke3d/gameexec.o
Built object obj/duke3d/gamevars.o
Built object obj/duke3d/player.o
Built object obj/duke3d/premap.o
Built object obj/duke3d/sector.o
.
.
.

- gekürzt, s. Bei Interesse in der Online-Ausgabe!
```

```
Built object obj/build/animvpx.o
Linked eduke32
Built object obj/duke3d/astub.o
.
.
.
Built object obj/build/startgtk.editor.o
Linked mapster32
Build started using:
compiler: "g++ -std=gnu++11 -fno-exceptions -fno-rtti  -O2  -fno-stack-protector
-funswitch-loops -fomit-frame-pointer -flto -fno-strict-aliasing -fjump-tables
-W -Wall -Wextra -Wpointer-arith -Wno-char-subscripts -Wno-missing-braces
-Wwrite-strings -Wuninitialized -Wno-attributes -Wno-strict-overflow -Wno-
unused-result -Wlogical-op -Wcast-qual  -funsigned-char -DNDEBUG -DUSING_LTO
-U_FORTIFY_SOURCE -D_FORTIFY_SOURCE=0 -DUSE_LIBPNG -DUSE_LIBVPX -DSTARTUP_WINDOW
-DHAVE_VORBIS -DHAVE_FLAC -DSDL_TARGET=2 -I/usr/include/SDL2 -D_REENTRANT
-DHAVE_GTK2 -pthread -I/usr/include/gtk-2.0 -I/usr/lib/x86_64-linux-gnu/gtk-
2.0/include -I/usr/include/gio-unix-2.0/ -I/usr/include/cairo
-I/usr/include/pango-1.0 -I/usr/include/atk-1.0 -I/usr/include/cairo
-I/usr/include/pixman-1 -I/usr/include/libpng12 -I/usr/include/gdk-pixbuf-2.0
-I/usr/include/libpng12 -I/usr/include/pango-1.0 -I/usr/include/harfbuzz
-I/usr/include/pango-1.0 -I/usr/include/glib-2.0 -I/usr/lib/x86_64-linux-
gnu/glib-2.0/include -I/usr/include/freetype2 -DRENDERTYPESDL=1 -DMIXERTYPESDL=1
-DUSE_OPENGL -DNOASM -DPOLYMER -Isource/build/include -Isource/mact/include
-Isource/audiolib/include -Isource/enet/include "
linker: "g++ -std=gnu++11 -fno-exceptions -fno-rtti   -O2  -fno-stack-protector
-funswitch-loops -fomit-frame-pointer -flto -fno-strict-aliasing -fjump-tables
-lm -lvpx -lrt -lSDL2_mixer -L/usr/lib/x86_64-linux-gnu -lSDL2 -ldl -pthread
-lpng -lz -lFLAC -lvorbisfile -lvorbis -logg"
-rwxr-xr-x 1 cbehrens cbehrens 2021712 Apr 30 17:19 eduke32
-rwxr-xr-x 1 cbehrens cbehrens 1533440 Apr 30 17:19 mapster32
```

Im Verzeichnis **eduke32** hat der Linker zwei ausführbare Dateien erstellt:

**eduke32**: Das ist die zum Spielen benötigte Programmdatei.

**mapster32**: Das ist – man ahnt es fast – ein Editor zum Bearbeiten und Erstellen eigener Level.

Auf den Editor komme ich vielleicht später noch zurück.

## 4. Einrichten und Erweitern des Spiels

Den Quellcode benötigen wir nun nicht mehr, daher habe ich die beiden Dateien in das Verzeichnis

**~/Spiele/eDuke32**

verschoben. In das Verzeichnis muss vor dem Start auch noch die oben bereits erwähnte Datei DUKE3D.GRP kopiert werden. Die Programme lassen sich dann dort entweder per Doppelklick oder – etwas eleganter – von der Konsole aus dem Verzeichnis heraus mit dem Befehl

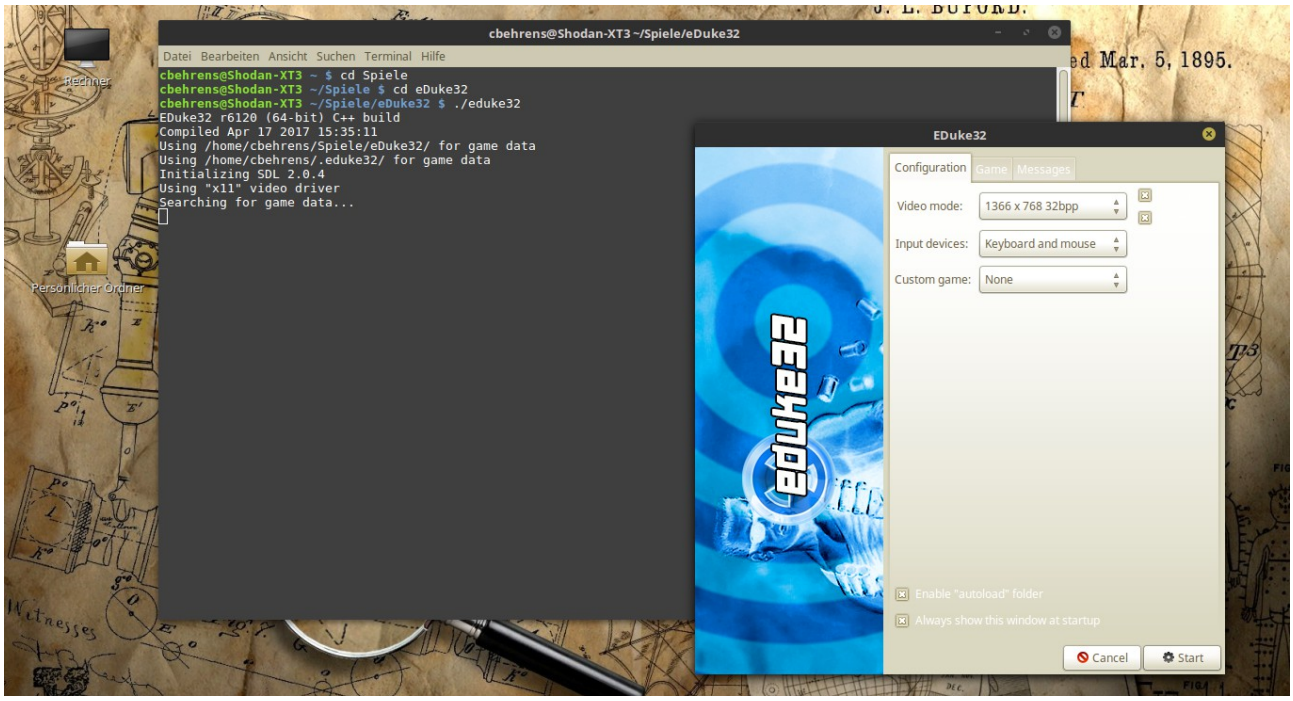
```
$ ./eduke32
```

starten. Der Start aus der Konsole ist vor allem immer dann sinnvoll, wenn dem Programm beim Aufruf noch bestimmte Parameter übergeben werden sollen.

Mit dem Konsolenbefehl

```
$ ./eduke32 -map [file.map]
```

z. B. können eigene Level geladen werden usw. Probiert das ruhig einmal aus, das sollte dann etwa so aussehen:




Unter „Messages“ nennt das Programm die Verzeichnisse, in denen es nach den Spieldaten sucht:

```
Using /home/cbehrens/Spiele/eDuke32/ for game data
Using /home/cbehrens/.eduke32/ for game data
```

Bevor wir mit dem Spielen beginnen, sollten wir das Spiel aber erst noch etwas zeitgemäß „aufbohren“. Dafür empfehle ich folgende Erweiterungen:

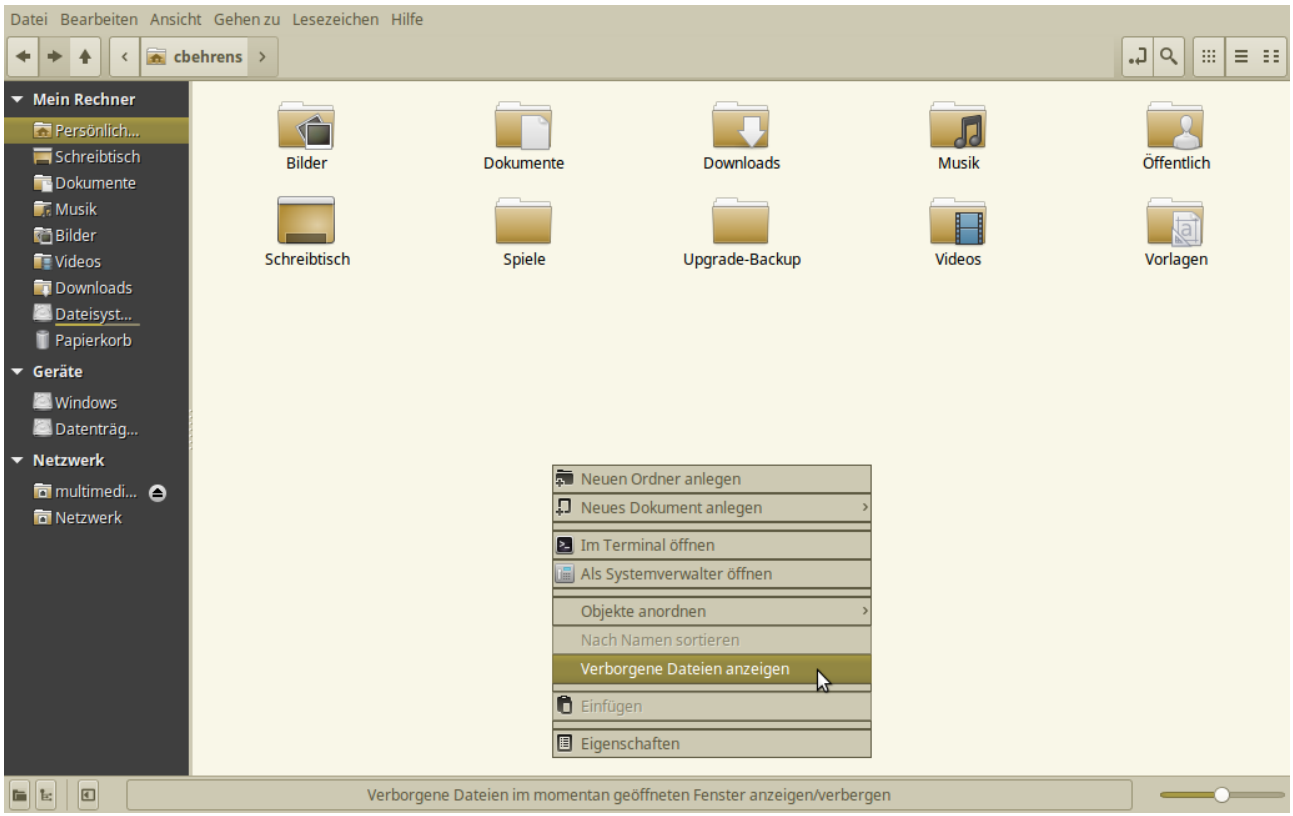
- **High Resolution Pack (HRP):** Ersetzt die doch etwas in die Jahre gekommenen Grafiken, vor allem Texturen durch neue, höher auflösende. Unter Linux wird das **Duke3D HRP v5.4 ZIPFILE (890 MB) - *Standalone HRP file*** benötigt. Als (optionales) Zubehör gibt es dann noch das **XXX Pack v1.33** - *by NightFright.* Damit lässt sich das Spiel dann noch etwas – äh – weniger jugendfrei gestalten als ohnehin schon.
- **Duke3D HRP Music Packs**: Polieren den Sound ordentlich auf.

Sämtliche heruntergeladenen ZIP-Archive bitte nicht entpacken, sondern so wie sie sind in das Verzeichnis **autoload** verschieben. Laut Meldungen sucht eduke32 zwar in beiden Verzeichnissen (s. o.), meiner Erfahrung nach sollte aber das Verzeichnis

```
$ ~/.eduke32/autoload
```

(also das versteckte Verzeichnis direkt im Heimatverzeichnis) verwendet werden.

*Hinweis: Unter Linux werden Dateien und Verzeichnisse mittels eines dem Namen vorangestellten Punktes „versteckt“. Der Dateimanager zeigt das .eduke32-Verzeichnis daher standardmäßig nicht an. Das lässt sich leicht ändern: Einfach mit dem rechten Mousebutton in einen freien Bereich klicken und im Kontextmenu den Punkt „Verborgene Dateien Anzeigen“ auswählen:*




*– und nicht erschrecken, da dürften einige Verzeichnisse erscheinen...*

Sollte das Verzeichnis **.eduke32** noch nicht vorhanden sein: Bitte einmal das Programm **eduke32** starten (s. o.), das erstellt dann das Verzeichnis, in dem es u. a. die Konfigurationsdateien, aber auch Spielstände und Screenshots (die können während des Spiels mit der Taste F12 aufgenommen werden) ablegt.

Sind alle ZIP-Archive im Verzeichnis **autoload** abgelegt, kann das Spiel bzw. eduke32 erneut gestartet werden. Angesichts der doch nicht unerheblichen Dateigrößen kann es dabei zu spürbaren Ladezeiten kommen. Da ich von einer SSD aus spiele, kann ich dazu nicht viel berichten, hier geht alles sehr schnell.

Wem das noch nicht ausreicht:

- **EDuke32 Addon Compilation v3.13 (635 MB)** - *by NightFright* (Details)

Und nun: Let‘s rock…!